

Home Based Cooperative Web Caching

Ming Zu

Jaspal Subhlok

Sui Shen

Yuan Zhang¹

Department of Computer Science, University of Houston, Houston, TX, 77204

ABSTRACT

Web caches in a cooperative cache array must be able to determine if another cache in the array has a specific URL. This is achieved by a broadcast query in the ICP Protocol. In CARP protocol a particular URL can only be cached in a designated cache. Summary cache and Hint cache maintain global knowledge of URLs cached in the array. A broadcast query is expensive and limits scalability, mapping each object to a specific cache eliminates locality, and maintaining global knowledge is a significant overhead. This paper proposes a new cooperative caching protocol called HOME which avoids these drawbacks. Caches in HOME store all objects but give preference to home objects. Every web object is a home object for one of the caches in the array based on URL hashing. When a proxy cache receives a URL request from a client, it first attempts to service it locally, and then contacts the home cache for the URL. We demonstrate with trace driven simulation that HOME locates objects in the array effectively and achieves performance on par with cooperative ICP arrays, but without the inter-cache queries associated with ICP query broadcasts.

Keywords: Web Caching, Internet Cache Protocol, Cooperative Caching.

1. INTRODUCTION

The explosive growth of the World Wide Web has motivated many techniques to improve the performance of Internet services. Web proxy caching is recognized as one of the most effective and practical ways to alleviate Internet traffic, lower client access latency and improve web server availability [2,5,17]. However, a single proxy cache can become the bottleneck and a single point of failure [9]. Deploying groups of cooperating caches address the robustness and scalability limitations caused by a single proxy cache and thus enhance the benefits of web caching [8,9,11,12,13]. Such a cache array can be organized in multiple ways [8,12], but for our discussion, we assume that it consists of a set of distributed, cooperating sibling caches. Every proxy cache serves a set of clients and a client directly communicates with a single cache, which is its local cache. Caches use sibling caches to service client requests when they do not have a requested document. A request for a document leads to a “local hit” if the local cache contains the document and a “remote hit” if another cache has the document. Otherwise it is a “miss” and the origin website of the document delivers the document.

Various cooperative web caching strategies have been proposed over the last decade, from the pioneering Harvest system [3] to several recent cache array schemes [4,11,15,16,19,23]. A key feature that distinguishes cooperative web caching schemes is the mechanism employed to determine whether a sibling cache contains a specific web object. In Internet Caching Protocol (ICP) [19], sibling caches are explicitly queried whether they

have a particular object or not. The main drawback of this method is that the workloads on caches as well as inter-cache traffic increase substantially due to queries. A different approach is taken in Cache Array Routing Protocol (CARP) [16], where every web object is assigned to a designated cache with a hashing technique based on its URL. CARP is a query-less protocol since a simple hash function can locate the only cache that could store a given URL, which also implies that there is no duplication of objects in the cache array. The main drawback of CARP is that a given cache is allowed to store only a fraction of web objects, hence its clients will see a low local hit ratio. This translates to high user latency since most objects must be obtained from remote caches or servers. Summary Cache [4] and Hint Cache [15] propose efficient mechanisms to maintain global knowledge of objects in the cache array. Both approaches dramatically reduce the multicast overhead associated with checking if a sibling cache contains a particular object, leading to a reduction in cache workload and inter-cache traffic. However, these methods introduce significant design complexity and network or storage overheads.

In this paper, we introduce a new cache array protocol called HOME. Proxy caches can store any object locally in HOME, but there is also a designated home cache for every object. If an object is in the cache array, it will almost certainly be in the home cache also. In case of a local miss, the cache only checks the home cache for an object, since it is extremely unlikely, although not impossible, that an object exists in the cache array but not in the home cache. This is the key feature of HOME. As compared to the ICP protocol, querying for an object across the whole cache array is eliminated at the cost of a very small probability of a *false miss* where an object is in the cache array but could not be found. HOME uses a hashing scheme to associate home caches with URLs like CARP, but caches can store other objects too and popularity of URLs is maintained globally. Local hit ratios are expected to be significantly higher than CARP while array hit ratios are expected to be slightly lower. These results are achieved by allowing caches to store all objects, *but favor home objects*. In general, popular objects will be replicated on all caches but less popular objects and some very large objects will be available only from the home cache. The specific methods used to achieve this biased storage policy are discussed in the paper.

In this paper we describe HOME cooperative caching protocol and discuss how it compares with ICP and CARP, two well-known protocols. Home is designed as an alternative to an all sibling ICP array or a CARP array, which are normally employed on a small scale. We compare the performance of HOME, ICP and CARP under a variety of conditions and parameters based on simulation with cache usage logs obtained from National Laboratory for Advanced Networking Research (NLNR) [10]. The results demonstrate that HOME can achieve performance that is similar or slightly better than ICP,

¹ Yuan Zhang is currently with Williams Gas Pipeline Transco

An expanded version of this paper is available at http://www.cs.uh.edu/~mzu/mingpaper_final.pdf

while considerably reducing inter-cache traffic and workload associated with ICP queries. The main result is that HOME achieves the near-best performance with a very low overhead in terms of inter-cache traffic.

This paper is organized as follows. Section 2 describes the HOME protocol and compares it with other cooperative caching protocols. In section 3, we describe our trace-driven simulation environment. Section 4 contains a quantitative comparison of HOME, ICP and CARP. Section 5 investigates false misses in HOME. Section 6 contains conclusions.

2. HOME COOPERATIVE CACHING PROTOCOL

A HOME cache array consists of a group of cooperating proxy caches. Each user is assigned to a particular cache and sends all requests to this *local cache*. There is also a *home cache* associated with every web object. The identity of the home cache for a particular web object can be easily computed by applying a hash function to the URL. Caches can store home as well as non-home objects. However, if an object is larger than a fixed *threshold*, only the home cache is allowed to store the object. The procedure for handling a request for an URL sent by a client to its local cache is as follows:

The local cache determines whether the request is a local hit or a local miss by checking its storage:

Local hit: The local cache simply sends a copy of the object to the client.

Local miss:

1. The local cache identifies the home cache of the object by applying a predetermined hash function.
2. If the local cache is itself the home cache, it obtains the object from the web, sends a copy to the client, and keeps a copy in its storage. The processing is complete.
3. Otherwise, the local cache queries the home cache for the object.
4. The home cache determines if it is a hit or a miss by checking its storage and informs the local cache.
 - a. **Home Hit:** The local cache requests and receives the object from the home cache.
 - b. **Home Miss:** The local cache requests and receives the object from the origin server.
5. The local cache delivers the object to the client.
6. If the object size is less than a *threshold*, the local cache keeps a copy of the object. If the object was obtained from the server, *the local cache sends a copy to the home cache*.

The query from a local cache to a home cache is suitable for implementation over UDP, while the objects are transferred over TCP with HTTP requests. We assume that the caches follow a LRU replacement policy [1,20] although it is not fundamental to HOME.

We point out the main features of HOME. When a local cache does not contain a requested object, it only checks with the home, not the others. This saves the overhead of querying caches but opens the possibility that an object is stored in the array but not in the home cache. When a new object is obtained from the web, a copy is always sent to the home cache. Also, when a local cache obtains an object from the home cache, the

copy of the home object is refreshed automatically, that is, its age is reset to zero for the purpose of LRU cache replacement policy. These aspects of HOME imply that if an object is in the cache array, then the home cache will almost certainly have a copy, hence it is not worth checking the other caches.

An auxiliary feature of HOME is the existence of a *threshold*. If an object is larger than the *threshold*, it may still be stored in the home cache but not in the other caches. The reason is to prevent large objects from being replicated across the array. For our simulations, the *threshold* was set to 1Mbyte and hence this applies only to very large objects.

It is instructive to compare HOME with ICP [19], where all caches act independently except that they check other caches to serve misses, before going to the origin website. Caches in HOME favor home objects since home objects serve the whole array, while other objects only serve the local clients. It is expected that ICP will achieve somewhat higher local hit ratios. In ICP, all caches tend to have very similar sets of objects when a steady state is reached, while in HOME, the caches have a larger fraction of their corresponding home objects. Hence there is less duplication and we expect a higher array hit ratio for HOME. However, *false misses* that are possible in HOME can reduce the array hit ratio. Perhaps the most significant difference between ICP and HOME is that the number of queries to check if an object exists in another cache is dramatically lower in HOME. The number of queries generated in response to a single local cache miss equals the number of sibling caches in ICP, while in HOME only one query to the home cache is generated. This has significant impact on cache overhead and network traffic. Hence, if the performance of HOME and ICP were similar in other aspects, the reduction in overhead would be a strong reason to favor HOME.

We now compare HOME with CARP [16], a well-known protocol proposed by Microsoft. A web object can only be stored in a designated cache in CARP, hence there is no duplication of objects in the cache array. The local hit ratios are inherently low in a CARP array since only a fraction of the web objects can be cached on any single cache. CARP is primarily designed for the situation where all caches are part of a cache cluster and there is little difference between accessing a local cache and a remote cache. However, if a local cache can be accessed more efficiently than a remote cache, CARP yields poor performance because of a low local hit ratio. HOME borrows the basic idea of having a designated home cache for objects from CARP. However, HOME allows caching of all objects on every cache, which significantly improves the local hit ratio performance.

An alternative approach to nearly eliminating inter-cache queries is to maintain an index of objects in the cache array on individual caches. A refined form of this approach is introduced in the Summary cache protocol [4]. Hint Cache [15] is also based on individual caches exploiting global knowledge of the network and it specifically addresses performance across multiple level hierarchies. It is difficult to compare HOME with these approaches quantitatively but we shall argue that HOME can achieve similar results without the overhead of maintaining the metadata. Even though HOME caches do not maintain any explicit metadata, the caches implicitly share popularity and cooperate in object replacement, which has been pointed out as important components of performance [8].

HOME is designed for a moderate number of caches in a local area or metropolitan area setting, and not for a worldwide hierarchy. However, HOME protocol can be used at the leaves of a large-scale hierarchy, while another protocol is used across the higher levels of the hierarchy. Specifically, groups of siblings in an ICP hierarchy could benefit from HOME, while the interaction between parents and children remains unchanged. Direct extension of HOME to large-scale hierarchies is not addressed in this paper.

3. SIMULATION ENVIRONMENT

A trace-driven simulation program was developed to evaluate the performance of HOME protocol and compare it with other cache array protocols. The program simulates a cache array environment in which multiple caches handle incoming web requests. Cooperation among caches can be set to follow ICP, CARP, or HOME protocol. For ICP simulation, only an all sibling hierarchy is considered. Every cache contacts all its siblings on a cache miss before contacting the web server. For CARP simulation, it is assumed that every object can be stored in exactly one cache, and the relevant cache can be readily identified based on the object's URL. Both these protocols can be configured in other ways also but we consider this basic configuration most suitable as a reference for comparison. All results in this paper are based on a one-day log from National Laboratory for Applied Networking Research² (NLNR) that consisted of approximately 2.5 million requests, but we have verified that the results are similar for other days. Results are also qualitatively similar for another trace obtained from Digital Equipment [22] but are not included in this paper. We are currently planning validation experiments with additional traces including the UC Berkeley Web Proxy Trace [18].

In the simulation, each client has an assigned local cache. For CARP and HOME, each web object is also assigned a home cache based on its URL. Properties of the environment, such as number of caches, cache sizes, time to query a sibling cache, etc. are input as parameters to the simulation program. By varying the parameter settings, the program can simulate and compare the performance of different cache array protocols in different environments. The performance of different protocols is analyzed in the following six categories: local hit ratio, local byte hit ratio, array hit ratio, array byte hit ratio, average response time, and inter-cache queries.

In order to compute the average response time, we have to assign response times to objects that are delivered by the local cache, objects that are delivered through another cache in the array, and objects that are obtained from a web server. It is critical to have good estimates for the simulation results to be meaningful. Our simulation is based on traces and the traces do contain information about the time it took to serve an object request and whether the request resulted in a cache hit or a cache miss. However, an object request that resulted in a cache hit in the trace may result in a cache miss in our simulation, and vice versa. Hence the trace information is not sufficient.

The approach we have developed for estimating response times is based on our observation of the relationship between object sizes and cache hit and cache miss times in traces. Both cache hit response time and cache miss response time are modeled as linear functions of object sizes as follows:

$$T_{\text{response}} = \text{factor} \times S_{\text{object}} + \text{offset} \quad (1)$$

where S_{object} is the size of the requested object in bytes and *factor* and *offset* are constant coefficients that are determined empirically. The goal is to estimate the median³ response time of requests of different sizes. For the traces that we used, the optimal coefficients were estimated to be *factor* = 8.21 and *offset* = 29 for cache hit response time, and *factor* = 13.10 and *offset* = 157 for cache miss response time. Figure 1 compares the actual median values of the response times for objects of different size ranges, and those predicted by our linear function. It is clear that the two are in close agreement.

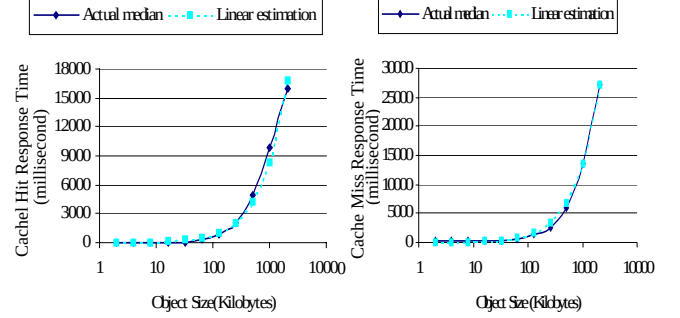


Figure 1: Estimation of response time for different object sizes

These response time estimation functions are used to assign response times to object requests that were serviced by the local cache as well as the requests that were serviced by a web server. We also have to estimate the response time for objects that were first obtained from a sibling cache and then sent to the client. The ratio between the time to deliver an object from a local cache and the time to deliver the same object from a sibling cache is a parameter labeled *beta* in the simulation. *beta* reflects the additional time that is needed to retrieve the object from a sibling cache and depends on both the network between caches and the network between a client and a cache. In our simulations, the default value for *beta* is set to 1.5, which reflects a 50% overhead in obtaining an object from a sibling cache versus a local cache. We also present results with different values of *beta* where appropriate. The response time also depends on the time it takes to complete an inter-cache query, which is another parameter in simulation. Typically such queries are sent on top of UDP transport protocol. It is assumed that all inter-cache queries are successfully received and replied.

The simulator adopts LRU for all protocols, which is one of the most commonly used and efficient cache replacement algorithms [1,6,20]. Freshness of documents is not simulated. Removal of objects from a cache is decided solely by cache capacity and the LRU replacement policy.

4. PERFORMANCE COMPARISON OF ICP, CARP AND HOME

We compare the performance of HOME with ICP and CARP for different cache sizes, different numbers of caches, and different ratios of array hit response times to local hit response times. For all results in this section, the inter-cache query time is 10 milliseconds. The *threshold* (the largest object size that

² The date of the log file is March 19th, 2002.

³ The mean was not used because a few large (and possibly erroneous) values in the log were found to have a strong influence on estimation.

can be stored locally) is set to 1Mbytes. The size was selected heuristically to avoid duplicating very large objects in the array.

Performance with different cache sizes

For this set of experiments, there are six proxy caches, and the size of each cache is varied from 40 Mbytes to 280 Mbytes, which reflects an overall cache array capacity from 2% to 15% of the total size of distinct requested objects. The *beta*, which is the ratio between the time to service an object request from a local cache versus the time to service that request from another cache in the array, is set to 1.5 for this simulation. The results are presented in Figures 2 to 5.

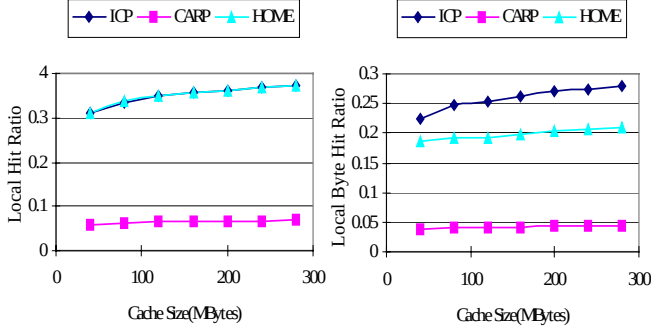


Figure 2: Local hit ratios and local byte hit ratios

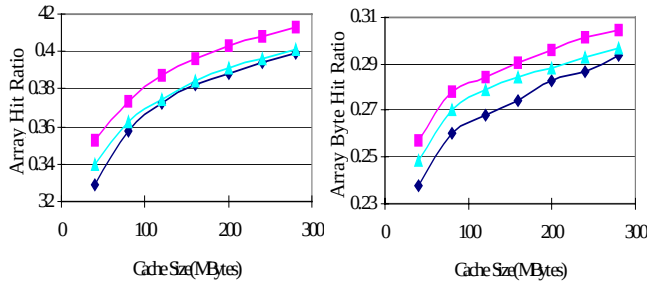


Figure 3: Array hit ratios and array byte hit ratios

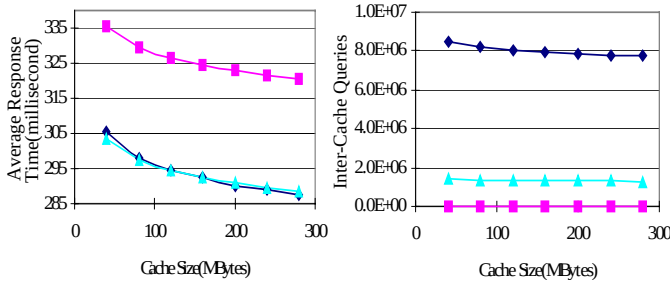


Figure4: Average response time Figure5: Inter-cache queries

Figure 2 shows that HOME and ICP have much better local hit ratios than CARP. This is no surprise since an individual CARP cache is allowed to store only a fraction of all objects. HOME and ICP have virtually the same hit ratio, but ICP has a better byte hit ratio. The reason is that HOME has a threshold where very large objects are only stored on home caches not others, and they can substantially affect the byte hit ratios.

Figure 3 shows that CARP has the best array hit ratio and array byte hit ratio. This is expected since CARP allows no duplication of objects and therefore stores the maximum number of distinct objects in a cache array. HOME has a slightly better array hit ratio and a clearly better array byte hit

ratio than ICP. The reason is that HOME has less duplication and more distinct objects than ICP in the cache array.

Figure 4 shows that HOME and ICP have considerably lower response times than CARP. The reason is the low local hit ratio of CARP that we discussed earlier. HOME and ICP exhibit similar response time performance with HOME faring slightly better for smaller cache sizes. The lower local byte hit ratio of HOME is compensated by the higher array hit ratio and array byte ratio resulting in similar average response time. Figure 5 shows that the number of inter-cache messages generated by HOME is dramatically lower than the number generated by ICP. Upon a local miss, ICP generates inter-cache queries whose number equals the number of proxy caches in the array, while HOME generates only one inter-cache query per local miss.

The main result is that the response time performance of HOME is qualitatively similar to ICP but the number of inter-cache queries for HOME is lesser by an order of magnitude. HOME is able to achieve similar, or even better, performance than ICP without having to check every sibling cache for objects through a query broadcast on every miss. The lightweight scheme for object location proposed in HOME that requires checking only the home cache appears to be effective.

Performance with different numbers of caches

For this set of experiments, individual cache size is fixed at 80M bytes and the number of caches is varied from 4 to 8. Other parameters are unchanged from the simulation with different cache sizes. The results are presented in Figure 6.

We observe that changing the number of caches does not impact the relative local hit ratios among ICP, CARP and HOME significantly. Array hit ratios for all protocols improve with the number of caches, but the array hit ratio of HOME improves faster than ICP. One reason is the high duplication of content in ICP. Adding more caches adds relatively small number of new objects in the arrays since most of the contents of an added cache would already exist in the cache array. It is important to note that the array hit ratio of HOME keeps improving with increasing number of caches implying that checking just the home cache is effective for checking the availability of objects even in larger cache arrays.

The average response time for HOME decreases as more caches are added and the rate is slightly faster than ICP. At the same time, the traffic generated by ICP increases rapidly as the number of caches increases, while the traffic generated by HOME stays virtually unchanged. This is expected given that HOME generates one inter-cache request per local cache miss, while ICP generates as many requests as the number of caches in the array on every local cache miss. This demonstrates that HOME protocol is scalable, and its advantage over ICP increases with more caches in the array. It is important to note that this simulation does not take into account the performance impact of inter-cache traffic and overhead of processing inter-cache queries. As these favor home, the performance advantage of HOME is expected to be higher in actual deployments.

Performance with different speed inter-cache networks

An important aspect of a cache array environment is the performance of the network connecting the sibling caches. If all caches are on the same high-speed local area network, then a user may see little difference between an object served by its

local cache and one by the way of a sibling cache. Even though an object necessarily goes through an extra network hop when a local cache has to obtain it from a sibling cache before delivering it to a client, the performance impact can be negligible. One reason is that when a cache obtains an object from another cache, it is usually in the main memory not disk storage when it is sent to a client. The performance of memory hits is typically an order of magnitude better than secondary storage hits for web caches.

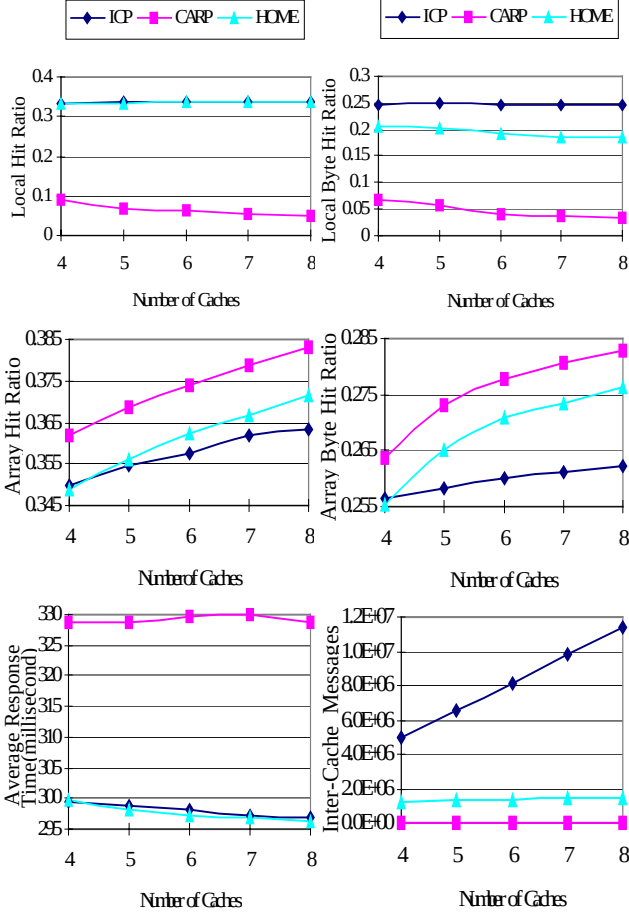


Figure 6: Performance with different numbers of caches

In our simulation, the inter-cache network performance is represented by the parameter β that determines the latency in obtaining an object through a sibling cache in relation to a local cache. If β is one, it means that a client experiences the same response times whether the object is found in its local cache or a sibling cache. A β of 2 means that the client-perceived latency is twice as much if an object is in a sibling cache versus a local cache. We compare the performance of CARP, ICP and HOME for values of β varying from 1 to 2. The number of caches is 6 and each cache has a capacity of 80Mbytes for this simulation. The results are plotted in Figure 7. Note that only the response time depends on β , and not hit ratios, so only the response time is plotted.

We observe that CARP has the best average response time for $\beta = 1$ but the response time increases rapidly as β increases. This is expected since a CARP client gets most of its data from sibling caches not the local cache. The performance of HOME and ICP is similar over the entire range of β representing different network conditions. The conclusion is

that HOME performance stays stable with increasing β despite the fact that it gains more of its performance from array hits. We should also point out that in this simulation, the time for inter-cache queries is fixed and it is assumed that no inter-cache queries or responses are lost. In practice, on a slow or congested network, inter-cache queries will be slower and more likely to get lost, which is likely to deteriorate the performance of ICP more than HOME.

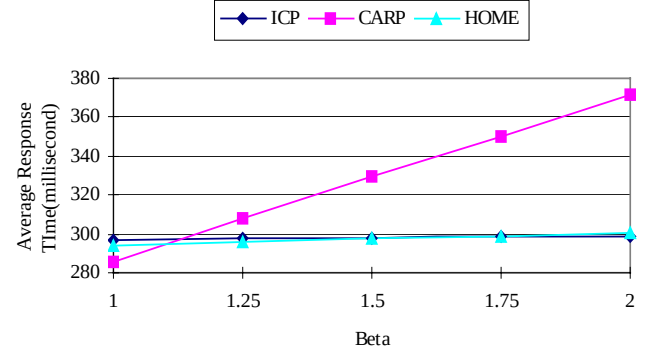


Figure 7: Average response time with different β

5. FALSE MISS PERFORMANCE

An interesting aspect of HOME is that *false miss* is a possibility and we explain how that can happen. A local cache receives a request from a client and it does not have that object. The local cache, in turn, contacts the home cache of the object, and the home cache does not have the object either. In such a situation, the local cache will obtain the object from the web server. However, it is possible that a sibling cache other than the home cache does have the object. This would be a case of a false miss since the object did exist in the cache array but was not found.

We have argued earlier that the probability of a *false miss* is very low. The reasons are as follows. First, whenever a cache obtains an object from a web server, it must deliver a copy to the home cache also. Second, since caches always contact the home caches for locally missed objects, the copies of objects in the home cache are likely to keep getting refreshed and live longer. Finally, very large objects can only be stored in the home cache. Actually, we found out in the experiments that false miss ratio, which is defined as the percentage of array misses that are false misses, is very small for HOME, varying between .15% and .3%, which shows that false misses are not a serious problem for HOME protocol.

6. CONCLUSIONS

This paper introduces a new protocol for cooperative web caching called HOME. Every web object is assigned a home cache in the HOME protocol. Caches give preference to home objects but they store all objects. In case of a local miss, only the home cache of an object is queried. The design goal of this approach is to allow individual web caches to store all objects but eliminate the broadcast queries necessary to locate objects in a cache array. Another goal is to reduce the duplication of objects by making different caches prefer different objects. Using trace driven simulation, we demonstrate that HOME indeed achieves these goals and delivers good performance.

Trace driven simulation was used to compare HOME with ICP and CARP, the two most common protocols for cooperative

caching. HOME achieves better performance than CARP in most situations primarily because HOME caches can store all objects leading to significantly higher local hit ratios. The main result of this paper is that the performance of HOME is similar to ICP even though HOME generates an order of magnitude fewer inter-cache queries than ICP. Simulations show that this result holds across different cache sizes, different number of caches, and different network conditions. We are basically able to eliminate broadcast queries in ICP without paying a penalty in other aspects of performance. In this paper we have presented results from a single NLANR trace log, but we have verified that the results are qualitatively the same for other traces from NLANR and for one trace from Digital Equipment [22].

We stated earlier that only the home cache of an object is queried in case of a local miss in HOME protocol. This opens up the possibility of false misses, where an object is in the cache array but was not discovered by the protocol. Simulations show that the false miss ratio is very low for HOME and is not likely to be a significant performance factor. We also designed a version of HOME called Refreshing-HOME that further reduces the false miss ratio by one third or more with a small increase in overhead. This is described in the expanded version of this paper [24].

HOME is designed as a communication protocol for a moderate number of cooperating sibling caches in a local area or a metropolitan area setting. It offers an effective and lightweight mechanism for a cache to benefit from sibling caches in such environments. For a large worldwide hierarchy such as that maintained by NLANR, it can be employed at the leaf layer with existing protocols for higher layers. A complete hierarchy based on HOME is a subject of current research.

7. ACKNOWLEDGEMENT

This research was sponsored by the Texas Advanced Technology Program under grant number 003652-0424, and by the Texas Learning and Computation Center. We thank Dr. Martin Herbordt at Boston University and Mr. Kevin Leigh at Compaq for their advice and help.

REFERENCES

- [1] M. Arlitt, C. Williamson, "Trace Driven Simulation of Document Caching Strategies for Internet Web servers", **The Society for Computer Simulation SIMULATION Journal**, Vol. 68, No. 1, pp. 23-33, January 1997.
- [2] G. Barish and K. Obraczka, "World Wide Web Caching: Trends and Techniques", **IEEE Communications Magazine**, Vol.38, No. 5, pp.178-185, May 2000.
- [3] C. Bowman, P. Danzig, D. Hardy, U. Manber, and M. Schwartz, "The Harvest information discovery and access system", **Computer Networks and ISDN Systems**, Vol.28, no.1-2, pp.119--125, 1995.
- [4] L. Fan, P. Cao, J. Almeida, and A. Broder. "Summary cache: A scalable wide-area web cache sharing protocol", In **Proceedings of SIGCOMM '98**, September 1998.
- [5] S. Glassman. "A Caching Relay for the World Wide Web", In **the 1st International WWW Conference**, 1994.
- [6] M. Korupolu and M. Dahlin. "Coordinated Placement and Replacement for Large-Scale Distributed Caches", **Proceedings of IEEE Workshop on Internet Applications**, pages 62-71, July 1999.
- [7] T. Kroeger, J. Mogul, and C. Maltzahn. "Digital's Web Proxy Traces". August 1996. [online].
- [8] K. Lee, S. Sahu, K. Amiri, and C. Venkatramani, "Understanding the Potential Benefits of Cooperation among Proxies: Taxonomy and Analysis", **IBM Research Report**, RC22173 (W0109-015), September 2001.
- [9] R. Malpani, J. Lorch, and D. Berger, "Making world wide web caching servers cooperate", In **4th International WWW Conference**, December 1995.
- [10] "National Lab of Applied Network Research (NLANR)". <http://ircache.nlanr.net/>.
- [11] S. Paul and Z. Fei. "Distributed caching with centralized control". In **Proceedings of the 5th International Web Caching and Content Delivery Workshop**, May 2000.
- [12] M. Raunak. "A Survey of Cooperative Caching".
- [13] A. Rousskov and V. Soloviev. "On Performance of Caching Proxies", In **Proceedings of Sigmetrics'98**, June 1998.
- [14] S. Shen, "Design and Trace Driven Analysis of a HOME-Bias Cache Array Protocol", **Master's thesis**. University of Houston, Dept of Computer Science. May 2002.
- [15] R. Tewari, M. Dahlin, H. Vin, and J. Kay. "Design considerations for distributed caching on the internet". In **Proceedings of the 19th International Conference on Distributed Computing Systems**, May 1999.
- [16] V. Valloppillil and K. Ross, "Cache Array Routing Protocol v1.0. Internet Draft". February 1998.
- [17] J. Wang, "A survey of web caching schemes for the internet". **ACM Computer Communication Review**, vol. 29, no. 5, pp. 36--46, 1999.
- [18] "Web Traces and Logs". <http://www.cs.uh.edu/~mzu/webtraces.html>.
- [19] D. Wessels and K. Claffy, "ICP and the squid web cache". **IEEE Journal on Selected Areas in Communication**, vol. 16, no. 3, April 1998.
- [20] S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E. Fox. "Removal Policies in Network Caches for World-Wide Web Documents". In **Proceedings of the SIGCOMM '96 Conference**, Palo Alto, CA, Aug. 1996.
- [21] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy. "On the scale and performance of cooperative web proxy caching". In **the 17th ACM Symposium on Operating Systems Principles**, December 1999.
- [22] Y. Zhang, "HOME Based Internet Cache Array Protocol". **Master's thesis**. University of Houston, Department of Computer Science. May 2002.
- [23] L. Zhang, S. Floyd, and V. Jacobsen. "Adaptive Web Caching". In **the 2nd NLANR Web Cache Workshop**, June 1997.
- [24] M. Zu, J. Subhlok, Y. Zhang and S. Shen, "HOME Based Web Caching". http://www.cs.uh.edu/~mzu/mingpaper_final.pdf